**המכללה האקדמית להנדסה בראודה**

**המחלקה להנדסת תוכנה - קורס מחשוב ענן**

**תרגיל בית 3 – עבודה בצוותים**

*scrum master - שחר דלאל.*

*frontend developer – יואב כץ.*

*backend developer – אביב כהן.*

*product manager – יעל קנטר.*

*UI – איריס קנטר.*

*QA – שחר וכילר.*

***חלק ראשון: הגדרת SDP – software development plan – איטרציה 2***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| acceptance test | Executor | Task | Requirement | Iteration |
|  |  |  |  | Iteration 2 |
| Verify that users are redirected correctly after login. | Frontend developer - Yoav Katz | Implement transition for users' login page. | **1. Transition between login pages** |  |
| Verify that admins are redirected correctly after login. | Frontend developer - Yoav Katz | Implement transition for admins' login page. |  |  |
|  | UI – Iris Kanter | Design login page UI for users. |  |  |
| Ensure all required fields are present and functional. | Frontend developer - Yoav Katz  UI – Iris Kanter | Create visually appealing and user-friendly registration page. | **2. Registration functionality** |  |
| Test user registration process to ensure accounts are created successfully and check exceptional end cases. | Backend developer - Aviv Cohen  QA – Shahar Vachiler | Develop backend logic for user registration. |  |  |
| Ensure message design effectively communicates registration status to users. | Frontend developer - Yoav Katz  UI – Iris Kanter | Create visually appealing and clear messages for registration success/error. |  |  |
| Verify search history is correctly saved and associated with user accounts. | Backend developer - Aviv Cohen | Implement logic to store search history data in the database. | **3. Search history functionality** |  |
| Confirm search history is displayed correctly and accessible to users. | Frontend developer - Yoav Katz | Display user's search history on the search page. |  |  |
|  | UI – Iris Kanter | Create visually appealing and user-friendly display for search history. |  |  |
| Test chatbot interaction on the search page. | Frontend developer - Yoav Katz | Ensure seamless integration of chatbot UI with search functionality. | **4. Chatbot integration** |  |
| Test chatbot responses and functionality. | Backend developer - Aviv Cohen | Develop backend processes to handle user queries and provide responses. |  |  |
|  | UI – Iris Kanter | Create visually appealing and intuitive interface for chatbot interaction. |  |  |
|  | UI – Iris Kanter | Create intuitive and visually appealing UI components. | **5. Index management functions** |  |
| Confirm all index management functions are operational. | Frontend developer - Yoav Katz | Develop frontend components for editing, deleting, and adding indexes. |  |  |
| Validate that the backend for index management perform as expected, retrieving index data, updating existing indexes, and deleting indexes. | Backend developer - Aviv Cohen  QA – Shahar Vachiler | Implement backend processes to handle index management requests. |  |  |
| Confirm indexes are successfully added to the database and contribute to improved search functionality. | Backend developer - Aviv Cohen | Populate database with appropriate indexes based on project needs. |  |  |
| Ensure all functionalities are thoroughly tested and meet requirements. | QA – Shahar Vachiler | Identify and report any bugs or issues encountered during testing. | **6. Testing and debugging** |  |
|  | Scrum Master- Shachar Dalal | Coordinate code review process. |  |  |
|  | Scrum Master- Shachar Dalal | Project closure. |  |  |

***חלק שני: סגירת הפרויקט***

1. תיק למתכנת:

לאחר שהתחברנו למסד הנתונים ואנחנו מתחילים לקלוט ולהעביר נתונים האתר שלנו מחולק לחמישה חלקים ונוסף לשאר הפונקציות הממומשות בכל חלק כל החלקים משתמשים בפונקציה fetch\_page.

**fetch\_page:**

The fetch\_page function performs a GET request to the provided URL. It utilizes headers defined internally to bypass potential restrictions from the web server, primarily by altering the User-Agent value to mimic a browser. If a successful response with a status code of 200 is received, the function returns the page content as HTML processed by the BeautifulSoup module. Otherwise, it returns None. In case of any exception, the function also returns None.

**החלק הראשון הוא איסוף הנתונים בו אנחנו משתמשים בפונקציות הבאות:**

**get\_all\_urls:**

The get\_all\_urls function extracts URLs from the provided BeautifulSoup object (soup) representing a webpage. It iterates through all tags in the soup and checks if they contain an 'href' attribute, indicating a hyperlink. If found, it retrieves the URL and parses it using the urlparse function to extract the domain and path segments. The function filters out unwanted URLs based on criteria such as domain (must contain 'sap.com') and path length (up to 3 segments). Additionally, it excludes URLs containing specific unwanted words or referring to certain countries. Finally, it returns a list of valid URLs meeting these criteria.

By filtering out unwanted words we reached a reasonable number of URLs of 924 links.

**החלק השני הוא ניתוח האינדקסים:**

**index\_words\_for\_all\_urls:**

The index\_words\_for\_all\_urls function retrieves a webpage using the fetch\_page function and then processes its content. If the page is successfully fetched and parsed into a BeautifulSoup object, it extracts all words from the text content using a regular expression. It then calculates the term frequency (TF) for each word and stores it in a dictionary. Next, it filters out stopwords from the list provided and updates the index dictionary for each word, associating it with the URL ID and its TF. If the word already exists in the index, the function updates its entry with the new URL ID and TF. If the word is not in the index, a new entry is created for it. Finally, the updated index dictionary is returned.

**store\_index:**

The store\_index function takes an index dictionary and stores it in a Firebase database. It iterates over each word and its corresponding data in the index dictionary. Then, it sends a PUT request to the Firebase database endpoint '/Sapshell/Index' to store the word and its associated data. If the index dictionary is not None, it proceeds with the storing process.

**compute\_Rank:**

The compute\_Rank function calculates the rank (TF-IDF) for each word in the index based on the term frequency (TF) and inverse document frequency (IDF). It first fetches the index from the Firebase database. Then, it iterates over each word and its corresponding data in the index. For each word, it retrieves the URL ID list and calculates the IDF based on the total number of documents. After that, it iterates over each URL ID and its TF dictionary, calculates the rank, and updates the TF dictionary with the computed rank. Finally, it stores the updated TF dictionary back into the Firebase database.

**החלק השלישי הוא תהליך החיפוש:**

**Search:**

The Search function receives a query and returns the documents that best match the query. It uses natural language processing techniques to parse the query, separate it into words, and transform the words into their root form (using the Stemmer). The function then looks up the words in an index which is a dictionary that maps words to the documents in which they appear.

The function returns the documents that contain all the words in the query, and sorts them according to a ranking calculated from the frequency of the words in the documents. The documents that contain more words than the query get a higher ranking. The function returns the ten highest ranked documents.

If no document contains the words from the query, the function returns `None' and prints a message that there are no results for the query. It is a complex function that combines techniques of natural language processing, information retrieval, and data science.

**החלק הרביעי הוא ניהול המשתמשים והאדמינים:**

**index\_word\_in\_db:**

The index\_word\_in\_db function queries the Firebase database to retrieve the existing index and all URLs data. It then iterates through a subset of URLs, fetching and processing each webpage's content using the fetch\_page function. For each webpage, it extracts all words and calculates the term frequency (TF) for the new word. If the TF is greater than zero, it constructs a URL ID list for the new word. Finally, it updates the index in the database with the new word and its associated URL ID list.

**compute\_Rank\_for\_word:**

The compute\_Rank\_for\_word function calculates the TF-IDF score for the specified word using the total number of documents in the database and the IDF value calculated from the index. It retrieves the URL ID list for the word from the database, calculates the IDF, and iterates over each URL ID, updating its TF dictionary with the computed rank. The updated TF dictionaries are stored back into the Firebase database.

**store\_index:**

The store\_index function iterates through the index dictionary and stores each word along with its associated data in the Firebase database.

**delete\_index\_from\_db:**

he delete\_index\_from\_db function removes the specified word from the index in the Firebase database.

**edit\_index\_in\_db:**

The edit\_index\_in\_db function first deletes the old word from the index and then calls the index\_word\_in\_db function to add the new word to the index.

**delete\_url\_from\_db:**

The delete\_url\_from\_db function deletes the specified URL from the Firebase database.

**add\_index\_in\_db:**

The add\_index\_in\_db function adds a new word to the index in the Firebase database. It checks if the word meets certain criteria, retrieves all URLs data, indexes the new word, stores the updated index, and calculates the rank for the new word.

**user\_exists:**

The user\_exists function checks if a user with the specified username exists in the Firebase database.

**create\_account:**

The create\_account function creates a new user account with the given username and password in the Firebase database.

**add\_search\_history:**

The add\_search\_history function adds a search item to the search history of the specified user in the Firebase database.

**login:**

The login function verifies the provided username and password against the stored user data in the Firebase database and logs in the user if the credentials match.

**show\_search\_history:**

The show\_search\_history function retrieves and displays the search history for the specified user from the Firebase database.

**fetch\_and\_process\_data\_for\_chart:**

The fetch\_and\_process\_data\_for\_chart function retrieves data from the Firebase database under the path '/Sapshell/Index'. It collects all word counts and processes them to identify rare words with only one occurrence. It then prepares data for a chart by extracting labels and values from the rare word counts. Finally, it returns a JSON string containing the labels and values for visualization.

**fetch\_indexes\_with\_most\_urls:**

The fetch\_indexes\_with\_most\_urls function fetches data from the Firebase database under the path '/Sapshell/Index'. It iterates over the indexes and counts the number of URLs associated with each index. It then prepares data for a chart by extracting labels and values from the index URL counts. Finally, it returns a JSON string containing the labels and values for visualization.

**החלק החמישי הוא הצאטבוט:**

**ShellGPT:**

The simulateResponse function serves as a chatbot for SAP-related inquiries. It analyzes the input message and generates appropriate responses based on predefined patterns.

For basic greetings and inquiries about the bot's identity, it responds with friendly messages introducing itself as SAP Bot.

For humorous queries, it delivers witty responses, such as jokes or references.

When asked about SAP-related topics like SAP, ERP, HANA, Fiori, ABAP, SAPUI5, GUI, NetWeaver, SuccessFactors, or Hybris, it provides concise explanations or descriptions of each term.

For any other inquiries, it generates random responses encouraging users to ask about SAP-related topics.

The function ensures an engaging and informative chat experience for users seeking information about SAP technology.

We tried and built a better quality chatbot, but we had trouble integrating it into the site's systems, so we resorted to imitation.

1. תיק למשתמש:

יש באתר החיפוש שלנו מספר מסכים

**The login and registration screen:**

If the user is already registered, all he has to do is enter a valid username and password, click the login button and he will be automatically moved to the next screen.

If the user is not registered, he will have to click on the register button

There he will have to choose a username and password, if the username already exists in the system, an appropriate error will be displayed, if he chooses a password longer than 14 characters (our limit to prevent congestion) an appropriate error message will be displayed, in the past there was also an error message if he entered invalid characters into the password or username but Unfortunately, we had to download this test because the javascript did not cooperate with the python regarding the string.

In any case, after the user has registered, he will still have to click on the login button and connect with his new user.

If the user is of the admin type, he will go to one page, and if he is of the user type, he will go to a separate page.

We do not allow the registration of admins through the site, only through the database.



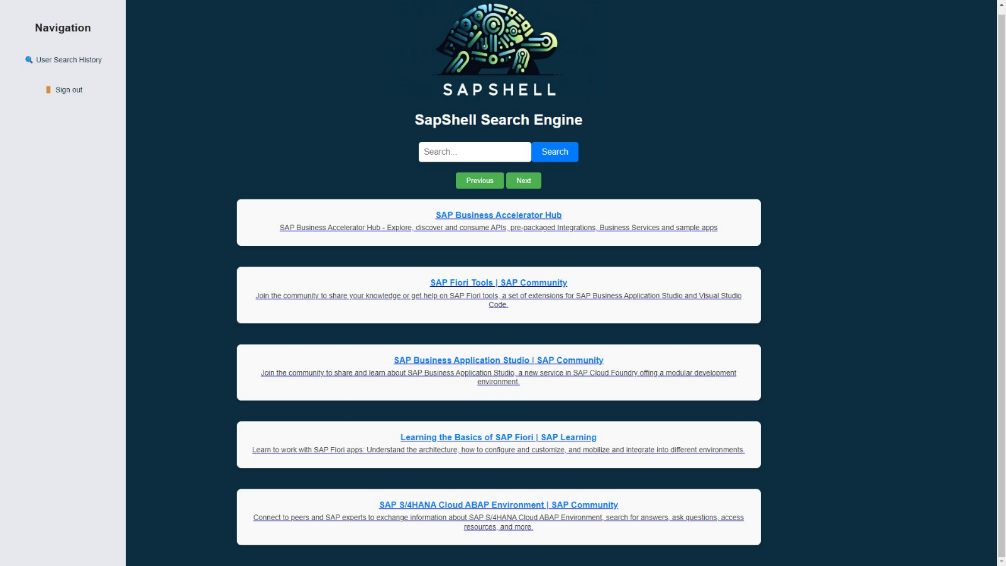
**Main user screen:**

When a user of the user type is connected to the system, he has two options: search and view search history, each of the options takes him to a dedicated page.



**search screen:**

When the user is on the search page, he has a search window, he can enter any query he wants and will receive an answer in the form of URL pages that match the search results, if no results are found, he will receive an appropriate message, if there are too many results, there is a page transition between the different results.



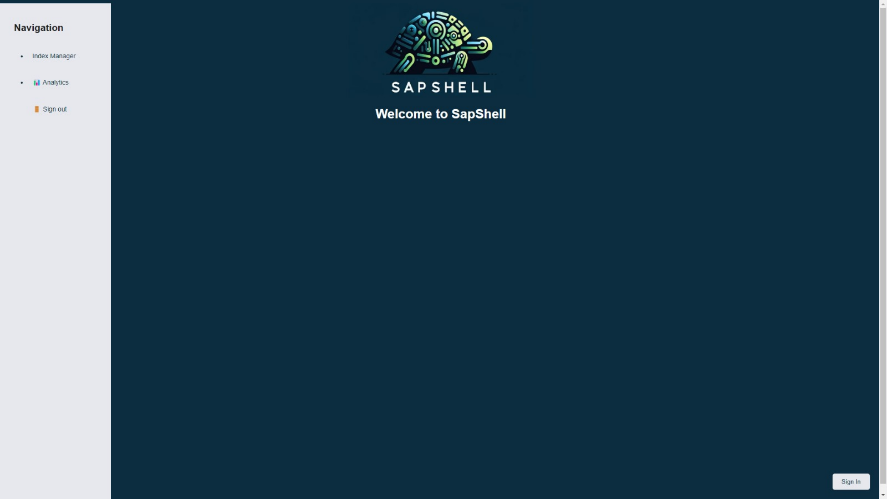
**show search history screen:**

When a user is on the show search history page, they will be shown all their old search results including the date and time the search was performed.



**Main admin screen:**

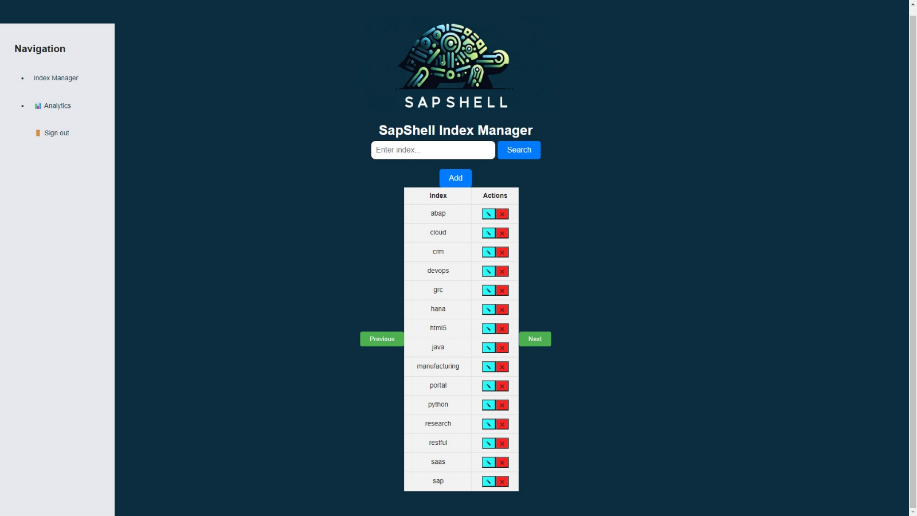
When an admin type user is connected to the system he has two options of his own: managing indexes and viewing statistics, each of the options takes him to a dedicated page.



**The index management screen:**

When an administrator is on the index management page, he can view, add, delete or edit indexes, the indexes will be shown to him in real time as they are in the database.

If he tries to add an index that already exists in the database he will be presented with an appropriate error.



**Statistics viewing screen:**

When an admin is on the Statistics viewing page, he can view statistics about the site, currently the only statistics that exist there is a comparison between 6 uncommon or "rare" words and 6 common words according to the number of urls where the words appear.



לצורך בדיקה הכנו שתי משתמשים אחד מסוג אדמין ואחד מסוג משתמש שכבר קיימת לו היסטורית חיפושים, כמובן שאפשר ליצור משתמשים חדשים בעזרת כפתור ה registerולהוסיף לו חיפושים באופן דינמי.

אדמין:

שם משתמש: shahar\_man95

סיסמא: abc12345

משתמש:

שם משתמש: yael123

סיסמא: 123

1. מצורף סירטון המתאר עבודה עם המערכת לשני סוגי המשתמשים שלנו.

<https://youtu.be/HpknW3cAo2w>